WHAT IS CLAIMED IS:

1. An interactive software engineering tool that, for distinct portions of a single unit of source code, presents a user thereof with behavior according to a corresponding set of lexical rules, wherein transition of the behavior from that in accordance with a first lexical context to that in accordance with a second lexical context is based on recognition of an opening boundary token according to the first lexical context and without use of a structural command to the interactive software engineering tool.

2. An interactive software engineering tool as recited in claim 1, wherein the behavior includes linguistically-driven typography.

3. An interactive software engineering tool as recited in claim 1, wherein the behavior includes lexical analysis of text based on a then operative one of the first and the second lexical contexts.

4. An interactive software engineering tool as recited in claim 1, wherein the distinct portions are delimited by the opening boundary token and a corresponding, automatically-added closing boundary token.

5. An interactive software engineering tool as recited in claim 1, wherein the first and second lexical contexts respectively correspond to one of:

a source language lexical context and a textual comment lexical context;
a source language lexical context and a string literal lexical context;
a source language lexical context and a character lexical context; and
first and second source language lexical contexts.

6. An interactive software engineering tool as recited in claim 1, wherein the single unit of source code is one of:

a line, statement or phrase;
a function, procedure or method; and
a markup language element,

thereof.

7. An interactive software engineering tool that, in response to introduction of a language-defined opening boundary token at a cursor position in an edit buffer, automatically inserts a corresponding closing boundary token, such that display of edit buffer content past the cursor position maintains its pre-introduction association with a first lexical context and with linguistically-driven typography therefor, while subsequent entry at the cursor position is subject to a second lexical context.

8. An interactive software engineering tool as recited in claim 7,
wherein display of symbols entered into the second lexical context is in accordance with linguistically-driven typography distinct from that employed in the first lexical context.

9. An interactive software engineering tool as recited in claim 7,
wherein lexical analysis of symbols entered into the second lexical context is in accordance with lexical rules distinct from that employed for the first lexical context.

10. An interactive software engineering tool as recited in claim 7,
wherein the second lexical context is delimited by the opening and closing boundary tokens.

11. An interactive software engineering tool as recited in claim 7, wherein the first and second lexical contexts respectively correspond to one of:
a source language lexical context and a textual comment lexical context;
a source language lexical context and a string literal lexical context;
a source language lexical context and a character lexical context; and
first and second source language lexical contexts.

12. A method of operating an interactive software engineering tool, the method comprising:
rendering a display presentation corresponding to a unit of source code, said display presentation corresponding to at least a first lexical context operative at an insertion point;

recognizing interactive entry of an opening boundary token at the insertion point; and

in response to said recognition of said opening boundary token, creating a second lexical context operative for subsequent interactive entry at the insertion point, wherein the second lexical context is delimited by said opening boundary token and a position in the source code immediately following the insertion point,

wherein said opening boundary token is a valid lexical token in accordance with one of the first and the second lexical context and not a non-lexical, structural command to the interactive software engineering tool.

13. A method as recited in claim 12, further comprising:

in response to said recognition of said opening boundary token, automatically inserting at said position in the source code immediately following the insertion point, a closing boundary token.

14. A method as recited in claim 12,

wherein stylistic rules applied to rendering of symbols within the second lexical context differ from those applied to rendering of symbols within the first lexical context.

15. A method as recited in claim 12,

wherein lexical rules applied to recognition of tokens within the second lexical context differ from those applied to recognition of tokens within the first lexical context.

16. A method as recited in claim 12,

wherein the first lexical context is a programming language lexical context;

wherein the second lexical context is string literal lexical context; and

wherein the opening boundary token is a quote (") character.

17. A method as recited in claim 12,

wherein the first lexical context is a programming language lexical context;

wherein the second lexical context is character lexical context; and
wherein the opening boundary token is a single quote (') character.

18. A method as recited in claim 12,
wherein the first lexical context is a programming language lexical context;
wherein the second lexical context is textual comment lexical context; and
wherein the opening boundary token is one of:
a multiple line comment token (/*);
a single line comment token (//); and
a document type comment token (/**).

19. A method as recited in claim 12,
wherein the first and second lexical contexts correspond to respective programming language lexical contexts.

20. A method as recited in claim 12,
wherein at least one of the first and second lexical contexts is a markup language lexical context.

21. A method as recited in claim 12,
wherein transitions between the first and second lexical contexts are performed in response to navigation events and in response to entry of valid lexical tokens such that the transitions are transparent to a user of the interactive software engineering tool.

22. A method as recited in claim 12,
wherein transitions between the first and second lexical contexts are performed in response to navigation events and in response to entry of valid lexical tokens such that a user of the interactive software engineering tool need not employ structural commands therefor.

23. A method as recited in claim 12, wherein the interactive software engineering tool includes one or more of:
an editor;

a source-level debugger; and

a source analyzer.

24. A method as recited in claim 12, wherein said unit of source code includes one or more of:

a line;

a statement;

a markup language element; and

a function or procedure.

25. A computer program product encoded in at least one computer readable medium and comprising:

functionally-descriptive encodings of at least first and second language contexts; and

instructions at least partially implementing a source code editor that invokes the second language context nested within the first language context based solely on recognition of a boundary token defined by the first language context and entered at the cursor position, while maintaining pre-existing language context past the cursor position.

26. The computer program product of claim 25, embodied as one or more of:

an editor;

a source-level debugger; and

a source analyzer.

27. The computer program product of claim 25, embodied, at least in part, as a language specialization component for integration with a software engineering tool.

28. The computer program product of claim 25, supplied, at least in part, via a communications medium for execution on a computer coupled thereto.

29. The computer program product of claim 25,

wherein the at least one computer readable medium is selected from the set of a disk, tape or other magnetic, optical, or electronic storage medium and a network, wireline, wireless or other communications medium.

30. A computer system comprising:

a display;

memory;

a language-based editor program executable thereby; and

a buffer defined by the source code editor program and instantiable in the memory,

wherein the language-based editor program renders contents of the buffer to the display in accordance with an associated language context, and

wherein the language-based editor program recognizes entry of a transitional opening token defined by a first language context and, in response thereto, associates text subsequently entered into the buffer at an insertion point thereof with a second language context, while maintaining a pre-existing association between the first language context and contents of the buffer past the insertion point.